|  |
| --- |
| ETL & Data Ware House |

|  |  |  |
| --- | --- | --- |
| Group member |  |  |
| Evan(Seung-Gyeom) KIM  Yingpeng YANG  Hongbo YE | DA ACC A03  DA ALT A03  DA ALT A03 |  |

TABLE OF CONTENTS

[I. INTRODUCTION AND DATA 3](#_Toc167045174)

[Call Data 3](#_Toc167045175)

[Call Types 3](#_Toc167045176)

[US States 3](#_Toc167045177)

[Employees 3](#_Toc167045178)

[Call Charges 3](#_Toc167045179)

[3 Steps ETL & Data Ware Housing 3](#_Toc167045180)

[II. STAGING DATABASE 4](#_Toc167045181)

[II-1.Call Data Table 4](#_Toc167045182)

[II-2. Call Types Table 8](#_Toc167045183)

[II-3. US State Table 8](#_Toc167045184)

[II-4. Employees Table 10](#_Toc167045185)

[II-5. Call Charges Table 11](#_Toc167045186)

[III. OPERATIONAL DATA STORE 12](#_Toc167045187)

[III-1. Call Charge Table 12](#_Toc167045188)

[III-2. Call Data Table 15](#_Toc167045189)

[III-3. Employees Table 22](#_Toc167045190)

[IV. DATA WARE HOUSE 25](#_Toc167045191)

[IV-1. Date Dimension Table 25](#_Toc167045192)

[IV-2. Employees Dimension Table 25](#_Toc167045193)

[IV-3. Call Data Fact Table 28](#_Toc167045194)

[V. Use case 33](#_Toc167045195)

[VI. CONCLUSION 37](#_Toc167045196)

# INTRODUCTION AND DATA

ServiceSpot, an IT company, needs an assistance to analyze their call center data. They provided their daily calls data hoping to improve their quality of service. The mission is to develop an EPL project with SSIS to conduct the data warehouse.

|  |  |
| --- | --- |
| Call Data Constitute 3 cvs files for 3 years of data 2018-2020 having same 6 columns   * + **CallTimestamp** : Date and Time   + **Call Type** : Type of calls   + **EmployeeID** : Foreign key form Employees table   + **CallDuration** : Counts by second   + **WaitTime** : Counts by second   + **CallAbandoned** : 0 and 1 Boolean value  Call Types  * + **CallTypeID** : 1-3   + **CallTypeLabel** : label of call types  US States  * + **StateCD** : 2-letter state code   + **Name:** Name of the state   + **Region:** East, West, etc  Employees  * + **EmployeeID** : Employ unique identifier   + **EmployeeName** : Full name of employee   + **Site** : Location of office   + **ManagerName** : Employee’s supervisor | Call Charges  * + **CallType** : label of call types   + **Call Charge** : charge per minute, 4 columns for each 4 years of charge evolution 2018-2021  3 Steps ETL & Data Ware Housing  * + **STA, Staging phase** :   Load and explore all the data   * + **ODS, Operational data store phase** : Clean and standardize the data. Create a “Technical\_Rejects” table in case of issue.   + **Data Ware House Phase :**   Organize one fact table. Create a “Functional\_Reject” table in case of issue. |

# STAGING DATABASE

At the data staging phase, the length is globally set to 255 as a precaution. It prevents errors when the data has an excessive length. The Code page is globally set to “1252 (ANSI – Latin I)” Since the “Data Flow Task”, the “OLE DB Destination” and the “Execute SQL Task” are iterative, they will be described only once.

## II-1.Call Data Table

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Remark :** In order to count the number of rows, “Get Data” function was used in Excel :

(Data 2018.csv, 33058 rows | Data 2019.csv, 32988 rows | Data 2020.csv, 32932 rows) The sum of rows counts is 98978. The difference between them is 3 (98978 – 98975 = 3). Assuming in SSIS, each table’s header row isn’t counted therefore, the result is correct.

* **Solution for 3 CVS files :** The Call Data covers 3 years of data, during 2018-2020. Each file contains one year of Call Data, making a total of 3 files. “Foreach Loop Container” will be used to gather 3 different tables into one table “CallData”. First, proceed with data staging as usual case, then add “Foreach Loop Container” in “Control Flow” to gather those files.
* **DFT - STA Call Data :** Drag and drop “Data Flow Task” SSIS Tool on “Control Flow” board.
* A screenshot of a computer

  Description automatically generated**FF\_SRC - STA Call Data :**

Drag and drop “Flat File Source” component into “Data Flow” **→**

Click “Browse” to select the file “Data 2018.csv” **→** Code page : 1252 (ANSI – Latin I), allows to load data without error and gives data type “varchar” **→** go to Advanced menu then set the OutputColumnWidth to 255

* **OLE\_DST - STA Call Data :** Drag and drop “OLE DB Destination” component in “Data Flow” **→** Connect to SQL Server → Create “CallData” table.  
  A screenshot of a computer code

  Description automatically generated
* **FELC- CallData2018-2020 :** Add a new variable “FilePath” and set the Data type “String”. As value, paste the path of the file “Data 2018.csv”. It sets the file path to search right file path when “Foreach Loop Container” will be executed.

A screenshot of a computer

Description automatically generated

Drag and Drop “Foreach Loop Container” component **→** Collection **→** Select “Foreach File Enumerator **→** Set the path of Folder **→** set the iteration reference as every csv format file in the FolderA screenshot of a computer

Description automatically generated

Variable Mappings **→** Select “User::FilePath”

A screenshot of a computer screen

Description automatically generated

Go to Properties setting for Flat File Connection Manager **→** Add an expression selecting “ConnectionString” then “**@[User::FilePath]**” as variable

A screenshot of a computer

Description automatically generated

* **SQL – Truncate STA Call Data :** Drag and drop “Execute SQL Task” SSIS component in “Control Flow”. Type the SQL Statement “TRUNCATE TABLE dbo.CallData”.

## II-2. Call Types Table

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

* **OLE\_DST - STA Call Types :**

A close-up of a computer code

Description automatically generated

## II-3. US State Table

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **OLE\_DST - STA US States :**

A close-up of a table

Description automatically generated

## II-4. Employees Table

A screenshot of a computer

Description automatically generated A diagram of a company

Description automatically generated A screenshot of a computer

Description automatically generated

* **OLE\_DST – STA Employees :**

A computer code with black and blue text

Description automatically generated

## II-5. Call Charges Table

A screenshot of a phone

Description automatically generated A screenshot of a computer

Description automatically generatedA screenshot of a data sheet

Description automatically generated

* **Remark :** The original Call Charges.csv file has 987 rows on the other hand, the essential data is only 3 rows. Despite of the empty rows, there was no impact on the result because the Lookup SSIS component doesn’t recognize the empty data. It matches only the rows where it can refer to the joinable keys.
* OLE\_DST – STA Call Charges :

A screenshot of a computer

Description automatically generated

# OPERATIONAL DATA STORE

Taking into account the anticipation for the 2021 data, the current focus revolves around the framework for processing new data through ETL (Extract, Transform, Load) process. The “Execute SQL Task” and “OLE DB Source” tanks are iterative, they will be described only once.

## III-1. Call Charge Table

Using the “Unpivot” to transform Call Charges table into operational form for ETL process.   
  
A screenshot of a computer

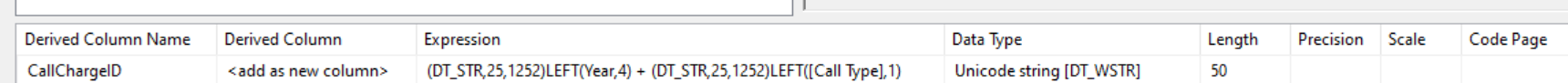
Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

* **OLE\_SRC STA – Call Charges :** Drag and drop “OLE DB Source” SSIS component → load Call Charges data from SQL Data base where the data of Call Chares is stored.
* **UPVT - Call Charges :** Drag and drop “Unpivot” SSIS component → Select the “Available Input Columns” → Destination column : “Call Charges” → Pivot Key Value : 2018, 2019, 2020 → Pivot key value column name : “Year”  
  A screenshot of a computer

  Description automatically generated  
  - Select the input column except 2021 (the 2021 data is not yet validated)  
    
  - If the Pass Through check box is marked, it will not be unpivoted but will instead be passed through to the output without any transformation.  
    
  - Destination Column refers to the column in the output data flow where the values from the unpivoted columns will be placed.  
    
  - Pivot Key Value refers to unique identifies for each rows in the unpivoted data. It differentiate between rows and determine which values belong to which original.   
  - Data Viewer  
  A screenshot of a computer

  Description automatically generated
* **DER - Create CallChargeID :** Drag and drop “Derived Column” SSIS component → Derived Column Name : CallChargeID , Expression : (DT\_STR,4,1252)LEFT(Year,4) + (DT\_STR,4,1252)LEFT([Call Type],1)  
    
  This table doesn’t have any joinable key, in this case, it’s necessary to create a new kink of primary key to join the fact table.  
  A screenshot of a computer

  Description automatically generated
* **DCNV – Resizing :** Drag and Drop “Data Conversion” SSIS component → Select the columns to resize the length   
  A screenshot of a computer

  Description automatically generated
* **OLE\_DTS – ODS Call Charges :**   
  A screenshot of a computer

  Description automatically generated

## III-2. Call Data Table

Pursuit to the STA phase :

1. “CallDuration” and “WaitTime” : will be cast into integer data type.
2. CallTimestamp : has a string data type and it’s similar to the datetime format such as “mm-dd-yyyy hh:mm”. Several data conversions will be anticipated. For instance, cast into datetime data type and separate into “yyyy-dd” and “hh:mm” creating two new different columns.
3. CallTypes : has only one byte of integer value. The Lookup SSIS component will allow obtaining the label of Call Types from “CallTypes” table. The data convention is not required, there is no issue joining with CallData type.
4. SLA Status : the standard call waiting time is the 35 scends. If the call was answered within 35 seconds, “Within SLA” will appear otherwise “Outside SLA” will appear.
5. EmployeeID : an adjustment of length will be applied.
6. CallAbandoned : will be cast into the Boolean data type.

**Part 1 : Data Conversion and Technical Rejects**Before joining CallTypes table with CallData table, cast the data type for “CallTimestamp”, “CallDuration” and “CallAbandoned” columns. At the end of this process, the redirected error columns will appear in the table named “Technical\_Rejects” on the ADM SQL database server.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

<for now, there is no error>

A computer screen shot of a code

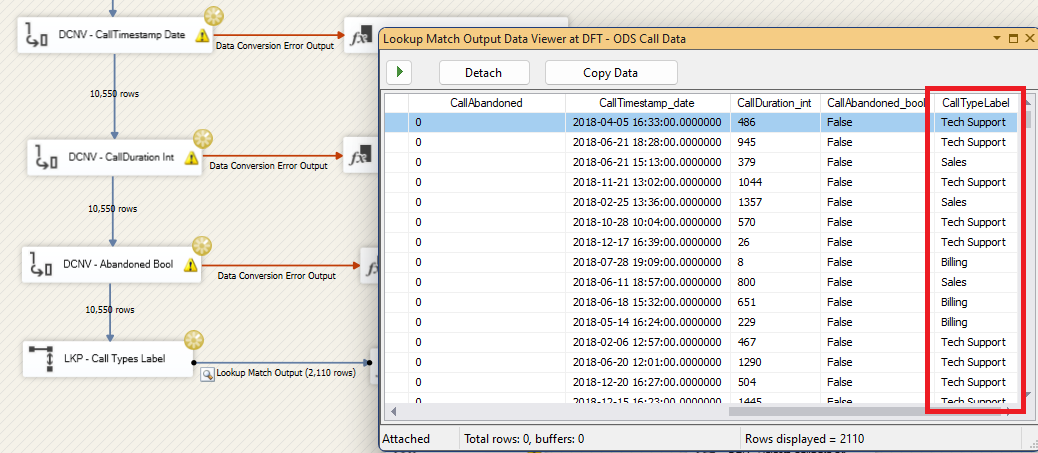
Description automatically generated with medium confidence

<Creation of “Technical\_Rejects” table in ADM SQL Server Database>

**Part 2 : Join the “Call Types” table with “Lookup” SSIS component.**

* **LKP - Call Types Label :** Drag and drop “Lookup” SSIS component in “Data Flow” → Connect to the SQL Server database where the STA Call Types table is stored. → Select the CallType table → Go to the Columns menu then link between Call Type and CallTypeID columns then mark the check box of CallTypeLabel.  
  A screenshot of a computer

  Description automatically generated
* Data Viewer



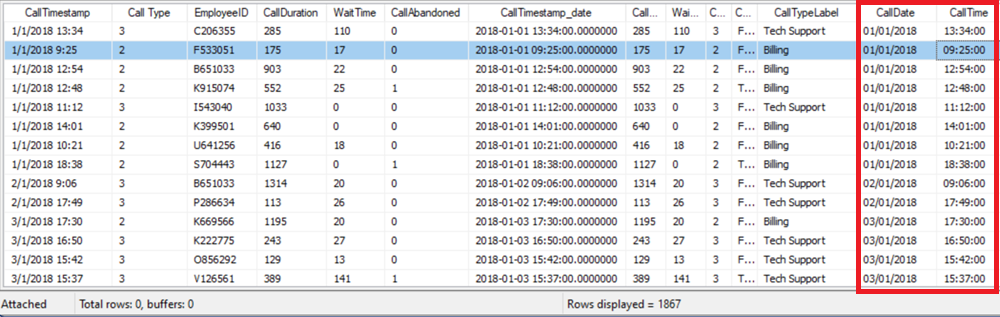
**Part 3 : Join the “Call Charges” table with “Call Data” table**

“CallChargeID” will be constituted by the “Year” (from CallTimestamp) and the first alphabet of the “CallTypeLabel”. For example, 2018B. For combining these two elements, the LEFT() function will be used. Currently, at this phase, the “CallTimestamp\_date” derived column has format “dd/mm/yyyy hh:ss.sssssss”. For the LEFT() function, the date format should be “YYY/DD/MM” then, the statement will be “LEFT(CallDate,4) + LEFT(CallTypeLabel,1)” which combines these two elements by slicing 4 letter from CallDate and 1 letter form CallTypeLabel from the left side.

* **Separate the CallTimestamp into CallDate and CallTime.**

First, arrangement of date format will be applied.

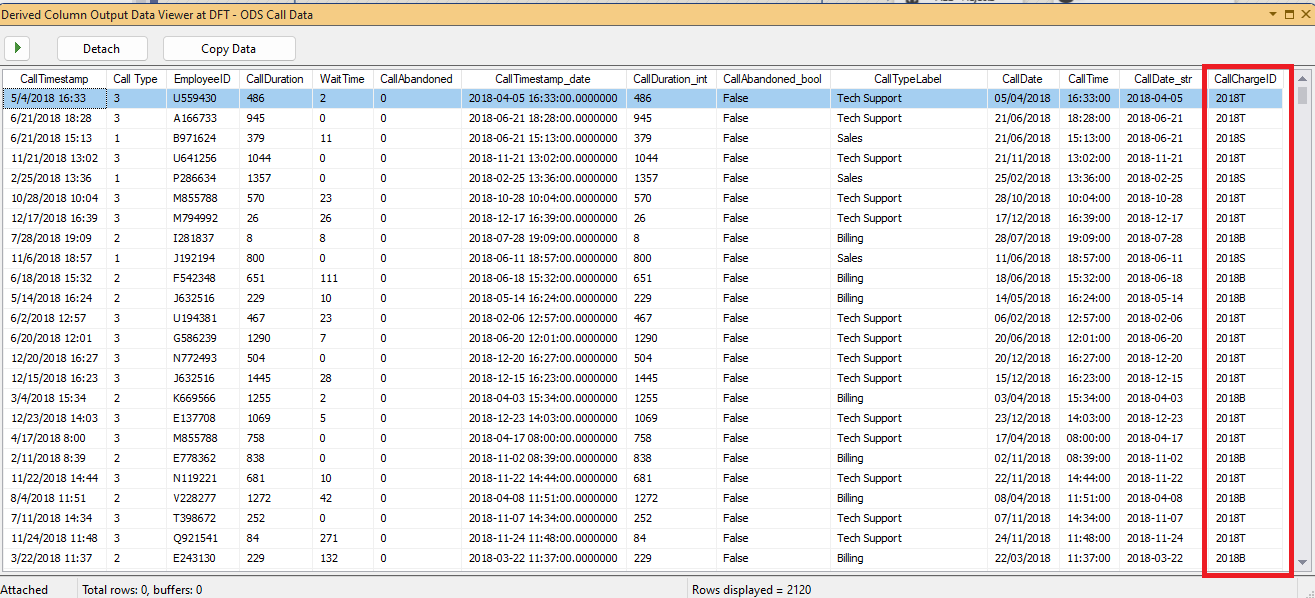
* **DER - CallDate and CallTime :** Select Lookup Match Output → Select CallTimestamp in “Available Input Columns   
  → Derived Column Name : CallDate , Derived Column : <add as new column>, Expression : (DT\_DBDATE)CallTimestamp\_date  
  → Derived Column Name : CallTime , Derived Column : <add as new column>, Expression : (DT\_DBTIME2,0)CallTimestamp\_date  
  - Data Viewer

****  
<the complexity of datetime format is solved>

* **Create a new CallChargeID and join the Call Data and Call Charges tables.**

As the LEFT() function requires a string data type, at first, cast CallDate into string data type then create a new CallChargeID column.

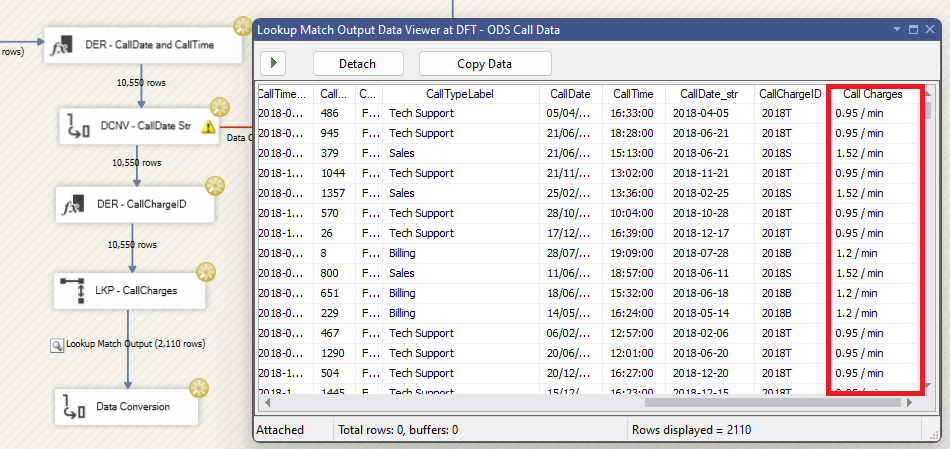
* **DCNV – CallDate Str :** Using the “Data Conversion” SSIS component, cast the data type of “CallDate” into a strig. As with the other data conventions, use the “Derived Column” SSIS component for the rejects and direct the error values to “OLE\_DST - AMD Rejects”
* **DER – CallChargeID :** Derived Column Name : CallChargeID, Derived Column : <add as new column>, Expression : (DT\_DBDATE) LEFT(CallDate\_str,4) + LEFT(CallTypeLabel,1)



* **LKP - Call Charges :** Select Lookup Match Output → Connect to ODS SQL server database and select the “CallCharges” table → Go to Columns menu → Link CallChargeId and CallChargeID then check the box for Call Charges

A screenshot of a call center

Description automatically generated



<Data Viewer>

* **DER – Remove min :** Derived Column Name : CallCharge\_remove\_min, Derived Column : <add as new column>, Expression : (DT\_DBDATE) LEFT(CallDate\_str,4) + LEFT(CallTypeLabel,1)

A screenshot of a computer

Description automatically generated

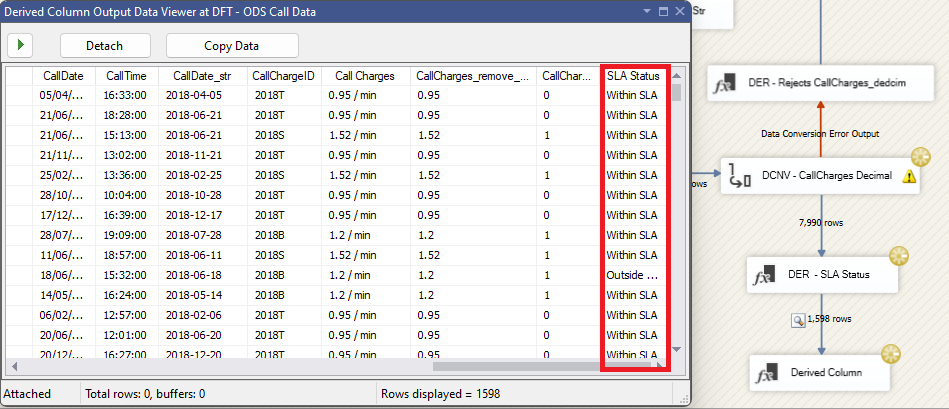
* **DCNV - CallCharges Decimal :** Cast data type into decimal then iterate same process for the “ADM Rejects”

A diagram of a software error

Description automatically generated

**Part 4 : SLA Status**

* **DER – SLA Status :** Derived Column Name : SLA Status, Derived Column : <add as new column>, Expression : WaitTime\_int < 35 ? "Within SLA" : "Outside SLA"



**Part 5 : ODS CallData**

* **DCNV – Resizing :** Reconfigure the length**A screenshot of a computer

  Description automatically generated**
* **OLE\_DST – ODS CallData :** Connect to SQL Server ODS database then select the “CallData” table and create a new “CallData” table → Mapping  
  A screenshot of a computer code

  Description automatically generated  
  A screenshot of a computer

  Description automatically generated

**Result “CallData” table :**

A screenshot of a table

Description automatically generated

## III-3. Employees Table

The Site column has a name of site and a State Code both. Dividing by State Code and the Site, Sate code will be joined with the US States table to have correct name of State and Region.

* Verify if there is any duplicate data

A screenshot of a computer program

Description automatically generated

<No duplicate>

* Data Flow for the ODS Employees data

A screenshot of a computer

Description automatically generated

* **DER – Site Code :**   
  A screenshot of a computer

  Description automatically generated
* **DER – Site :**



* **LKP – STA Us States :**   
  A screenshot of a computer

  Description automatically generated
* **DCNV – Resizing :**   
  A table with text and numbers

  Description automatically generated
* **OLE\_DST - ODS Employees :**A screenshot of a computer code

  Description automatically generated

A screenshot of a computer

Description automatically generated

# DATA WARE HOUSE

Beyond of the ODS phase, It should be ensured the quality of the data before working on the Data Ware Housing. First, add a Surrogate Key for each Dimension table in order to verify if there is missing value or errors. Second, build certain use case based on result of ETL tasks.

## IV-1. Date Dimension Table

* Create a Date Dimension Table based on the Dim Date code on SQL Server.

## IV-2. Employees Dimension Table

* **OLE\_SRC - ODS Employees** : Connect to the ODS SQL Server database then load ODS Employees table.
* **LKP – EmployeeID :** General menu → Select “Redirect rows to no match output” → Connection menu → Create a new “DimEmployees” table   
  A screenshot of a computer

  Description automatically generated  
  → Select the DWH SQL Server database and “DimEmployees” table → Only link the data between the “EmployeeID” and “EmployeeID” columns.
* “INT PRIMARY KEY Identity (1,1)” will be executed to create a new “DimEmployees” table. It incrementally generates the surrogate key of Employees data one by one according to (1,1) command.
* The “Redirect rows to no match output” setting redirect unmatched rows referring to the EmployeeID.
* **LKP - Any Change? :** Link “LKP – EmployeeID” into “LKP – Any Change?” then select “Lookup Match Output” → General menu → Select “Redirect rows to no match output → Connection menu → check the match columns except Input Column “EmployeeSurrKey” this column will get the Surrogate Key by previous scrip table to have created the “DimEmployees” table
* **CMD - Update Record :** Link “LKP - Any Change?” into “CMD - Update Record” and select Lookup Match Output → Connection Managers tap → Select the ODS SQL Server database → Component Properties tap → Write a script to update unmatched data   
  A screenshot of a computer

  Description automatically generated  
  → Mapp the columns  
  A screenshot of a computer

  Description automatically generated  
  - The Destination Column follows the order of generated table by “String Value Editor”.
* **OLE\_DST - DWH Employee :** Link “LKP – EmployeeID” into “OLE\_DST - DWH Employee” then select Lookup No Match Output.  
  A screenshot of a computer

  Description automatically generated

A screenshot of a computer

Description automatically generated

* Now the Dimension Employees table is ready to be using for some business purposes of analysis. Forming a “Star Schema” , it will be deployed with the Fact Table “Call Data”

## IV-3. Call Data Fact Table

For ensuring the quality of data on the Fact table, deployed SCD type 1 method. It calls “Slowly Changing Dimension”. It overwrites old data(Primary Key) with the new data(Surrogate Key) in dimension table.

* Join these two table “DimEmployees” and “FactCallData” with a Primary key “EmployeeId” and “EmployeeSurrogateKey”
* Check the missing values from the EmployeeID
* Make the outputs in case of the missing value and unmatched values referring this Primary key “EmployeeID”

**Part 1. Lookup EmployeeID and DateKey**

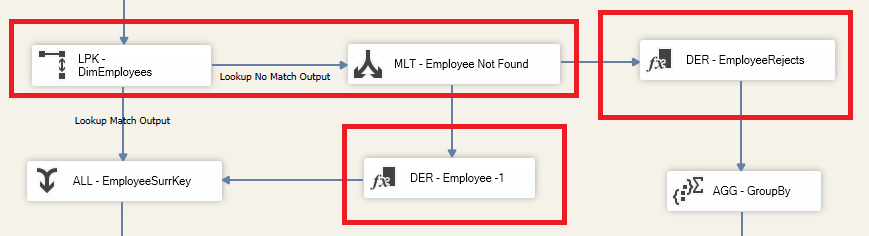
* OLE\_SRC - ODS CallData : Connect to the SQL Server database ODS CallData then load ODS CallData table.
* LPK – DimEmployees : make a relationship between “DimEmployees” table and “FactCallData” table.
* LKP – DimDate : make a relationship between “DimDate” table and “FactCallData” table. With “Date” column and “CallDate” column .
* OLE\_DST – FactCallData : Before preceding farther, observe the first table of “FacCammData”.  
  A screenshot of a computer code

  Description automatically generated A screenshot of a diagram

  Description automatically generated  
  A screenshot of a computer

  Description automatically generated
* The “EmployeesSurKey” generated without any issue. The “DateKey” contain historical records.

**Part 2. Searching missing values of “EmployeeID” and Aggregate group by Rejected values, then count their numbers. It is a important indicator to remark the duplicate data and the other unexpected issues.**

* **MLT - Employee Not Found :**   
  
* After “Multicast” SSIS component, two options will be applied one searching for the missing value through the two “Derived Columns” SSIS components, such as: “DER - Employee -1” and “DER – EmployeeRejects”. It facilitates automatization of SCD process.
* **DER - Employee -1 :** Derived Column Name : EmployeeSurrKey-1, Derived Column <add as new column> , Expression : -1

A screenshot of a computer

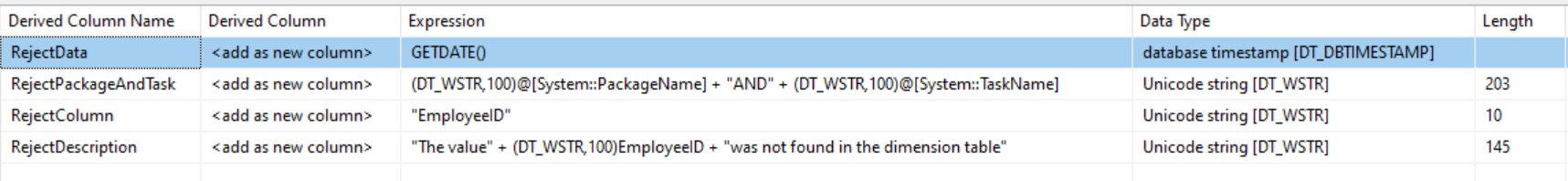
Description automatically generated

* This component will go to search the missing value
* **ALL – EmployeeSurrKey : “Union All” SSIS component receive the outcomes from “DER – Employee – 1“**

A screenshot of a phone call

Description automatically generated

**Part 3. Functional Rejects Table**

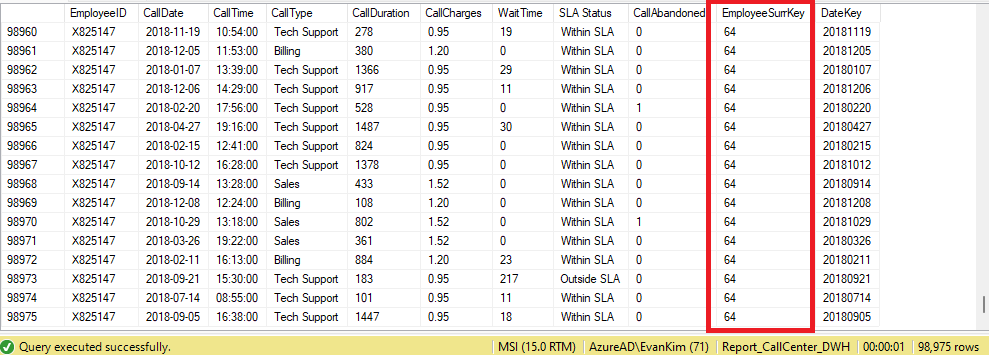
* **DER – EmployeeRejects :**   
  
* **AGG – GroupBy :**   
  A screenshot of a computer

  Description automatically generated
* Aggregate the Rejects outputs with their counting number.
* OLE\_DST – FunctionalRejects :   
  A screenshot of a computer code

  Description automatically generated  
  - As a precaution , set the maximum length to avoid the inconvenient cases for the data importing task.
* Result :

A screenshot of a computer

Description automatically generated



# Use case

1. **Manager Performance Evaluation**

SELECT [e]. [ManagerName],

COUNT([fc].[EmployeeID]) AS [TotalCalls],

SUM(CASE WHEN [fc]. [CallAbandoned]='1' THEN 1 ELSE 0 END) AS [AbandonedCalls],

ROUND(AVG(CAST([fc]. [WaitTime] AS FLOAT)), 2) AS [AverageWaitTime],

ROUND(SUM(CAST([FC].[Call Charges]\*[CallDuration]/60 AS FLOAT)), 2) AS [TotalCallCharges],

CAST((SUM(CASE WHEN [fc].[CallAbandoned] = '1' THEN 1 ELSE 0 END) \*1.0/ COUNT([fc].[Call Type])) AS DECIMAL(10, 3)) AS [AbandonRate]

FROM [dbo].[FactCallData] AS [fc]

INNER JOIN [dbo].[Employees] AS [e]

ON [fc]. [EmployeeID] = [e]. [EmployeeID]

GROUP BY [e]. [ManagerName]

ORDER BY [TotalCallCharges] DESC, [AbandonRate] ASC

**Objective:**

Evaluate call center performance metrics by manager.

**Key Metrics:**

TotalCalls: Total number of calls handled by each manager's team.

AbandonedCalls: Number of calls abandoned (where CallAbandoned = '1').

AverageWaitTime: Average wait time for calls, cast to float for accuracy.

TotalCallCharges: Total call charges, cast to float for accurate summation.

AbandonRate: Ratio of non-abandoned calls to total calls.

**Query Breakdown:**

SELECT: Calculates and retrieves the performance metrics.

FROM: Uses DWH\_FactCallData (fc) and DWH\_Employees (e) tables.

JOIN: Combines call data with employee/manager information on EmployeeID.

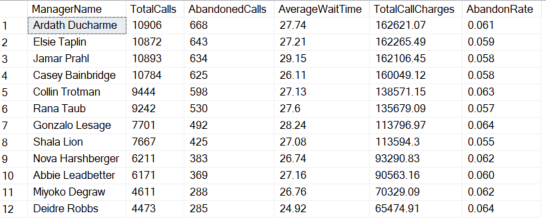
GROUP BY: Aggregates results by ManagerName.

ORDER BY : TotalCallCharges in descending order, and AbandonRate in ascending order.

**Usage:**

This query helps analyze the effectiveness of each manager's team in the call center by providing key performance metrics. It assists in identifying areas for improvement and making informed decisions

**Results:**



We can see that Ardath Ducharme is the manager that has the highest call charges (162K across three years), which Deidre Robbs is the one who performs worst.

1. **Site Performance Evaluation**

SELECT

[e].[Region],

[e].[Site],

COUNT(\*) AS [TotalCalls],

AVG([f].[CallDuration]) AS [AvgCallDuration],

SUM([f].[Call Charges]\*[CallDuration]/60) AS [TotalCallCharges],

SUM(CASE WHEN [f].[CallAbandoned] = '1' THEN 1 ELSE 0 END) AS [nb\_abandon],

CAST(SUM(CASE WHEN [f].[CallAbandoned] = '1' THEN 1 ELSE 0 END) / CAST(COUNT(\*) AS decimal(10,4)) AS decimal(10,4)) AS [AbandonRate],

SUM(CASE WHEN [f].[SLA Status] = 'Within SLA' THEN 1 ELSE 0 END) AS [nb\_within\_sla],

CAST(SUM(CASE WHEN [f].[SLA Status] = 'Within SLA' THEN 1 ELSE 0 END) / CAST(COUNT(\*) AS decimal(10,4)) AS decimal(10,4)) AS SLARate

FROM

[dbo].[FactCallData] [f]

JOIN

[dbo].[Employees] [e] ON [f].[EmployeeID] = [e].[EmployeeID]

GROUP BY

[e].[Region], [e].[Site]

ORDER BY

[TotalCalls] DESC;

**Objective:**

Evaluate call center performance metrics by region and site.

**Key Metrics:**

TotalCalls: Total number of calls handled in each region and site.

AvgCallDuration: Average duration of calls, providing insight into the handling efficiency.

TotalCallCharges: Total charges for calls, indicating the cost associated with the calls.

Nb\_abandon: the number of abandon calls.

AbandonRate: Ratio of abandoned calls to total calls, highlighting the rate at which calls are not completed.

Nb\_within\_SLA: the number of calls within SLA.

SLARate: Ratio of calls handled within the Service Level Agreement (SLA) to total calls, indicating compliance with service standards.

Query Breakdown:

SELECT: Calculates and retrieves the performance metrics for each region and site.

FROM: Uses DWH\_FactCallData (aliased as f) and DWH\_Employees (aliased as e) tables.

JOIN: Combines call data with employee information on EmployeeID.

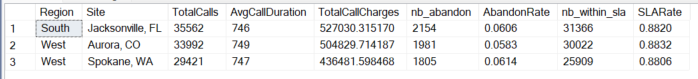
GROUP BY: Aggregates results by Region and Site.

ORDER BY: Sorts the results by the total number of calls in descending order to highlight the busiest regions and sites.

**Usage:**

This query helps analyze the performance of each region and site in the call center by providing key performance metrics. It assists in identifying high-performing areas and regions or sites that may need additional support or resources, thereby enabling informed decision-making to improve overall efficiency and service quality.

**Results:**



South region has the highest number of total calls and call charges.

1. **Year-on-year Evaluation**

WITH CTE AS (

SELECT YEAR([CallDate]) AS year

, COUNT(\*) AS nb\_calls

, CAST(SUM([CallDuration]\*[Call Charges]/60) AS INT) AS total\_charge\_dollar

FROM [CallCenter\_DWH].[dbo].[FactCallData] [f]

GROUP BY YEAR([CallDate])

)

SELECT year

, nb\_calls

, LAG(nb\_calls) OVER (ORDER BY year) AS nb\_calls\_last\_year

, nb\_calls - LAG(nb\_calls) OVER (ORDER BY year) AS nb\_calls\_yoy\_diff

, total\_charge\_dollar

, LAG(total\_charge\_dollar) OVER (ORDER BY year) AS total\_charge\_dollar\_last\_year

, total\_charge\_dollar - LAG(total\_charge\_dollar) OVER (ORDER BY year) AS total\_charge\_dollar\_yoy\_diff

FROM CTE

**Objective:**

To evaluate performance of each year and detect if there’s positive yearly growth.

**Methodology:**

A CTE (common table expression) is used and serves as an intermediate table to get the number of calls and total charges made for each year. Here a formular [CallDuration]\*[Call Charges]/60 is designed to calculate the charges made by each call.

Later in the output two new columns (last\_year and yoy\_diff) show up to compare and calculate between this year’s data and last year’s for both nb\_calls and total\_charge.

**Result:**

We can find that although the number of calls is declining year by year, but the total charge is increasing, which indicates the company is performing well.

1. **Power BI Dashboard**

A screenshot of a computer

Description automatically generated

After connecting the SQL server with Power BI dashboard we could make simple and real-time data analysis on the dataset. In simple conclusions, we can observe that January tends to be the busiest season in which the call centre gets the most profit compared to other months. Ardath is the manager who has the best performance in terms of the call charges surpassing others; and Ardath is also the employee who has the second the greatest number of calls. Other information and use cases needs could be fulfilled as well by the real-time monitoring in Power BI.

# CONCLUSION

We have made some choices during the ETL process:

**FactCallData**

This is our final fact table. We set it to be fact table because:

* It has quantitative data such as call duration, wait time and call charge, and contextual attributes such as call type, SLA status and call abandoned.
* It has transactional grain: each row in the table represents a single call, providing detailed, transaction-level data.
* It stores foreign keys to dimensional tables: EmployeeID and EmployeeSurrKey as foreign keys to dimensional table Employees.
* CallChargeID enables the the calculation of call charges with this table.
* CallTypeLable facilitates the categorization of each call records so that we could know to which type it belongs such as “Sales”, “Biling”, “Tech Support”, making it more straightforward for users.

It starts from table ODS\_CallData in the ODS step where we lookuped the tables CallType and CallCharges from STA and took the fields we think useful (call type and call charges). We combined it with our raw CallData table from the STA step. Nevertheless, as requested we also added the field SLA Status, which specify is if a call is within or outside SLA, in the ODS step and finally in DWS.

Therefore, this fact tale FactCallData that has every field concerning the calling data and considering its foreign keys, it can thus support various analyses, such as performance analysis, cost analysis and trend analysis, as what we have shown in the user cases.

**Employees**

It’s a dimensional table. The role of dimensional table is mainly to provide context to the quantitative data store in fact table, and we can find that our table Employees perfectly fits this role. It starts from the Employees table in STA, and we lookuped the geographical table, US States, to reduce complexity.

By embedding attributes such as StateName, Region, and Site directly within the DWH\_DimEmployees table, we ensure that all relevant geographic information is readily available alongside employee data. This integration facilitates quicker access to comprehensive datasets, enabling more efficient querying and analysis.

This approach not only simplifies our data model but also enhances the performance of our queries, reducing the complexity and potential for errors in our analyses. Consequently, analysts can now perform site-specific performance evaluations with greater ease and accuracy, supporting more informed decision-making across various organizational levels.

**Future expansion**

We believe that our project is a practical case of building an entire ETL process from scratch. If we have call data for future years (year 2021, 2022 etc.), we can just easily add them into the folder in the STA Call Data step, and the whole pipeline will still be running well.